TITLE:

# Approximating Vertex Cover on Dense Graphs (Evolutionary Advancement in Fundamental Theories of Computer Science)

AUTHOR(S):

Imamura, Tomokazu; Iwama, Kazuo

CITATION:

Imamura, Tomokazu ...[et al]. Approximating Vertex Cover on Dense Graphs (Evolutionary Advancement in Fundamental Theories of Computer Science). 数理解析研究所講究録 2004, 1375: 49-60

ISSUE DATE:

2004-05

URL:

http://hdl.handle.net/2433/25575

RIGHT:

京都大学学術情報リポジトリ
KURENAI 紅
Kyoto University Research Information Repository

数理解析研究所講究録 1375 巻 2004 年 49-60



# Approximating Vertex Cover on Dense Graphs

Tomokazu Imamura and Kazuo Iwama
School of Informatics, Kyoto University, Kyoto 606-8501
{imamura,iwama}@kuis.kyoto-u.ac.jp

今村 友和　　岩間 一雄
京都大学情報学研究科, 京都市
{imamura,iwama}@kuis.kyoto-u.ac.jp

**Abstract**

Although many problems in MAX-SNP admit a PTAS for dense graphs, that is not the case for Vertex Cover, which is MAX-SNP hard even for dense graphs. This paper presents a randomized approximation algorithm for Vertex Cover on dense graphs which is probably optimal: Let $\varepsilon = \overline{d}/\Delta$ where $\overline{d}$ and $\Delta$ are the average and maximum degrees of a given graph $G$. $G$ is said to be dense if its $\overline{d}$ is $\Omega(n)$. (i) Our algorithm achieves an approximation factor of $2 - \frac{\varepsilon}{1+\varepsilon/2}$ for dense graphs, which improves the best-known bound by Karpinski and Zelikovsky. (ii) It achieves the same factor for a wider range of graphs, i.e., for the graphs whose $\Delta$ is $\Omega(n\frac{\log\log n}{\log n})$.

## 1 Introduction

It is often true that although there is no good algorithm for general instances, there does exist an excellent algorithm for an important subclass. A typical example of such a subclass is *dense graphs* with an average degree of $\Omega(n)$. Recently, it has been shown [1] that a variety of NP-hard optimization problems, such as Max Cut and Steiner Tree, admit polynomial time approximation schemes (PTASs) for dense graphs, while the existence of such schemes for general instances of these problems would imply P = NP.

Dense graphs obviously constitute an important subclass since, information theoretically, almost all graphs are dense. Therefore we are naturally interested in whether or not other popular optimization problems, like Vertex Cover, have a similar property. Unfortunately, it is highly unlikely that Vertex Cover has a PTAS for dense graphs since the problem is still MAX-SNP hard for dense instances [4]. Nevertheless, it is an interesting question whether or not Vertex Cover can be approximated within a factor of $\alpha$ which is significantly smaller than two. The reason is much the same as before: For general instances, whether or not Vertex Cover has a $(2-\varepsilon)$-approximation algorithm is one of the major open questions and many researchers conjecture negatively. Karpinski and Zelikovsky [10] affirmatively answered this question with the algorithm whose approximation factor is $\frac{2}{2-\sqrt{1-\overline{d}/n}}$ where $\overline{d}$ is the average degree of the given graph. Unfortunately, however, this approximation factor quickly approaches to 2.0 while density decreases; it remains unanswered whether it is possible to take more substantial advantage of the density condition.

**Our Contribution.** Against this curiosity, this paper answers positively. Let $\varepsilon = \overline{d}/\Delta$. We present a randomized approximation algorithm which, with high probability, yields an approximation factor of $2 - \frac{\varepsilon}{1+\varepsilon/2}$ and runs in polynomial time if $\Delta$ is $\Omega(n\frac{\log\log n}{\log n})$. Thus the algorithm is quite attractive when $\varepsilon$ is close to one. For example, it approximates within a factor of 1.334 if the graph is regular, which remains true even for "quasi-dense" graphs whose degree is sub-linear in $n$ (but the computation time increases within polynomial). We can achieve a similar factor for random graphs if they satisfy the same density condition. Note that the approximation factor of the previous algorithm [10] gets close to 2.0 even for regular graphs if their density decreases.

Our algorithm depends on the same basic idea as the previous one by Karpinski and Zelikovsky [10]. Although it includes nontrivial extensions using recursion and randomization, the technique itself is fairly standard. Even so, our algorithm is probably optimal as mentioned above, which would indicate that it is relatively easy to take full advantage of the density condition in the case of Vertex Cover. This finding is another important contribution of this paper.

**Previous work.** For general Vertex Cover, finding any maximal matching in the given graph and choosing the all vertices in the matching yields a factor-2 vertex cover. However, no approximation algorithm of factor $2-\varepsilon$ is known for any constant $\varepsilon > 0$. The current best approximation algorithm was found by Halperin [7] whose approximation factor is $2 - \frac{2\ln\ln n}{\ln n}$. And all the known algorithms' approximation factor converge to 2 when $|V|$ is sufficiently large.

However, for the restricted version of the problems, many good approximation algorithms exist. For graphs whose maximum degree is bounded by $\Delta$, Hochbaum [9] presented $2-\frac{2}{\Delta}$ factor using a coloring method. Exploiting Halldósson and Radhakrishnan's [6] approximation algorithm for the Independent Set problem, one can obtain factor of $2-\frac{\log\Delta+O(1)}{\Delta}$. Using semidefinite programming, Halperin [8] asymptotically improved these results by presenting an approximation algorithm of factor $2-(1-o(1))\frac{2\ln\ln\Delta}{\ln\Delta}$. For small $\Delta$, Berman and Fujito [3] achieved a factor of $2-\frac{5}{\Delta+3}$. For planar graphs this problem admits PTAS [2]. Karpinski and Zelikovsky [10] showed that there is an approximation algorithm of factor $2/(2-\sqrt{1-\varepsilon})$ for dense graphs. They also showed that for everywhere-dense graphs, i.e., for graphs whose minimum degree is at least $\varepsilon|V|$, we can obtain an algorithm of factor $2/(1+\varepsilon)$.

For the negative direction, Håstad [8] showed that it is NP-hard to approximate Vertex Cover within a factor of less than $7/6 = 1.1666$. Dinur and Safra [5] improved this lower-bound to $10\sqrt{5}-21 = 1.3606$. Even for graphs whose maximum degree is bounded by $\Delta \geq 3$, this problem is still APX-complete and for large $\Delta$ it is NP-hard to approximate within 7/6 [4]. In [4], it is shown that Vertex Cover is Max-SNP hard (and hence does not have a PTAS) even for dense graphs. As a most recent result, Khot and Regev [11] showed that, if a certain conjecture about the PCP theory is true, we can prove the nonexistence of $2-\varepsilon$ factor approximation algorithm for Vertex Cover.

## 2 Notation and Basic Ideas

The *Minimum Vertex Cover* problem (MVC) is, for a given graph $G = (V, E)$, to obtain a set $C$ of vertices (called a *vertex cover*) such that (i) $C$ contains at least one endpoint of every edge and (ii) $|C|$ is as small as possible. $Opt(G)$ denotes the size of a smallest vertex cover and if an algorithm $A$ always finds a vertex cover $C$ such that $|C|/Opt(G) \leq r$, we say that $A$ achieves an *approximation factor* of $r$. We usually require that $A$ runs in polynomial time.

Our notations are standard: the number of vertices ($= |V|$), the average degree ($= 2|E|/|V|$), and the maximum degree are denoted by $n$, $\overline{d}$ and $\Delta$, respectively. $N(v)$ denotes the set of the neighbors of the vertex $v$ and in this paper, $N(v)$ does not include $v$ itself. $\deg(v)$ denotes $|N(v)|$. If we need to show the underlying graph $G$ explicitly, we use the notations such as $n_G$, $\overline{d}_G$, $\Delta_G$, $N_G(v)$ and $\deg_G(v)$.

In [10], Karpinski and Zelikovsky found the following simple and elegant property about a minimum vertex cover $C$: Suppose that we can take $k$ vertices $v_1, v_2, \ldots, v_k$ such that $\deg(v_i) \geq k$ for all $1 \leq i \leq k$. Then either (i) $\{v_1, v_2, \ldots, v_k\} \subseteq C$, or (ii) $N(v_i) \subseteq C$ for some $1 \leq i \leq k$, is true. (Reason: For any $v \in V$, either $v \in C$ or all vertices in $N(v)$ are in $C$. Hence the negation of (ii), i.e., $N(v_i) \not\subseteq C$ for all $i$, implies (i).) Therefore, if we consider the family of the $k+1$ sets, $\{v_1, v_2, \ldots, v_k\}, N(v_1), N(v_2), \ldots, N(v_k)$, then at least one of them, say $W$, is included in $C$. Furthermore, suppose that $k = \Omega(n)$. Then it turns out that we can yield an approximation factor of strictly less than two, by taking $W$ as a part of the vertex cover and then applying the standard 2-approximation algorithm [10] against the remaining graph. (Note that we do not know which $W$ is a part of $C$. So, we have to check all the $k+1$ possibilities and take a best result.) The question

**Algorithm** DVC-Apx$(t, i, G)$
1. **if** $i < t$ **then**
2. let $H(G) = \{v \in V(G) \mid \deg(v) \geq r(G)\}$
3. pick up $2(\log n)^2$ vertices from $H(G)$ uniformly at random and let $U_G$ be the selected vertices.
4. let $\mathcal{V}_G = \{H(G)\} \cup \{N(v) \mid v \in U_G\}$
5. **foreach** $V_j$ **in** $\mathcal{V}_G$, $1 \leq j \leq 2(\log n)^2 + 1$ **do**
6. select $r_i$ vertices from $V_j$ uniformly at random and let $V_j'$ be the selected vertices.
7. $C_j := V_j' \cup$ DVC-Apx$(t, i+1, G - V_j')$.
8. **end**
9. **return** $\min\{C_1, C_2, \ldots, C_{2(\log n)^2+1}\}$.
10. **else if** $i = t$ **then**
11. apply $\mathcal{VC}_2$ to $G$ and let $C$ be the resulting vertex cover.
12. **return** $C$.
13. **end**

Figure 1: algorithm DVC-Apx

is how we can achieve such a large $k$. Fortunately, it is quite easy if the graph is dense: Sort the vertices by their degrees and take $u_1, u_2, \ldots, u_k$ from the largest one until we have $k > \deg(u_k)$. Then a simple calculation guarantees that $k = \Omega(n)$ if $\bar{d} = \Omega(n)$, thus [10] gives the algorithm whose approximation factor is $\frac{2}{2-\sqrt{1-\bar{d}/n}}$.

Our new algorithm is an extension of this algorithm based on the following idea: (i) $\bar{d}/n$ in the approximation factor can be replaced by $\bar{d}/\Delta$. (ii) If the original graph is dense, then the remaining graph $G'$ after removing the vertex set $W$ is still somewhat dense. Hence we can use the same algorithm for $G'$. If we repeat this procedure $t$ times, then we can take more vertices in $C$ than before, but at the same time, our search space becomes larger, i.e., from $O(k) = O(n)$ previously to roughly $O(n^t)$. (iii) To reduce this search space, we introduce randomization. Instead of considering the $k+1$ sets, $\{v_1, v_2, \ldots, v_k\}, N(v_1), N(v_2), \ldots, N(v_k)$, we consider only $2(\log n)^2 + 1$ sets, $\{v_1, v_2, \ldots, v_k\}, N(v_{i_1}), N(v_{i_2}), \ldots, N(v_{i_{2(\log n)^2}})$. Here, the last $2(\log n)^n$ sets are selected uniformly at random from the original $k$ sets. Intuitively speaking, if some $v_{i_j}$ does not belong to $C$, then we have $N(v_{i_j}) \subseteq C$ and there is no problem. Otherwise, if all $v_{i_1}, v_{i_2}, \ldots, v_{i_{2(\log n)^2}}$ belong to $C$, then it follows that almost all vertices in $\{v_1, v_2, \ldots, v_k\}$ are in $C$ with high probability, which is again desirable for us. Note that the search space is reduced to $O((\log n)^{2t})$.

## 3 New Approximation Algorithm

Before starting arguments, we define a function $\gamma(G)$ as:

$$\gamma(G) = \begin{cases} \frac{\bar{d}n}{2\Delta} & (1 - \frac{\Delta}{n} \geq \frac{\bar{d}}{2\Delta}) \\ n - \sqrt{\Delta(2n-\Delta) - \bar{d}n} & (1 - \frac{\Delta}{n} < \frac{\bar{d}}{2\Delta}) \end{cases}$$

This function represents how many vertices our algorithm can extracts from the given graph $G$.

Figure 1 shows our algorithm DVC-Apx$(t, i, G)$. Our algorithm has a recursive structure and its maximum level of the recursion is given by integer $t \geq 0$. For a given graph $G = (V, E)$, we first call DVC-Apx$(t, 0, G)$. If $t$ is set to 0, then the algorithm is equal to the standard 2-approximation algorithms which is denoted by $\mathcal{VC}_2$. Otherwise it recursively calls DVC-Apx$(t, 1, G')$ for some $G'$.

Before explaining DVC-Apx$(t, i, G)$, we need to define the functions $r(G)$, $deg(i)$ and sequences of numbers $\{n_i\}, \{\bar{d}_i\}$ and $\{r_i\}$. Recall that our key operation is to take high-degree vertices

(denoted by $u_1, u_2, \ldots, u_k$ in the previous section). $r(G)$ determines the minimum degree of those vertices and is defined as follows:

$$r(G) = n_G \left(1 - \sqrt{1 - \frac{\bar{d}_G}{n_G}}\right).$$

In our algothrm, we repeat removing vertices which are included in a minimum vertex cover and, in doing so, we have to estimate the decrease of the sum of the degrees. For that purpose we define $deg(i)$ as

$$deg(i) = \min\{\Delta, n - i\}.$$

This represents the upper bound of the degree of $i$th removed vertex since, all vertices have degree at most $\Delta$ and $i$th removed vertex has degree at most the number of remaining vertices minus 1, which is $n - i$.

$n_i$, $\bar{d}_i$ and $r_i$ denote the numbers of vertices, a lower bound of the average degree, a lower bound of $r(G)$ of a graph $G$ and the sum of the number of removed vertices at the recursion level $i$. They are defined as follows:

$$\begin{aligned}
n_{i+1} &= n - s_i, \\
\bar{d}_{i+1} &= \frac{\bar{d}n - 2\sum_{k=1}^{s_i} deg(k)}{n_{i+1}}, \\
r_{i+1} &= \frac{\bar{d}_{i+1}}{1 + \sqrt{1 - \frac{\bar{d}_{i+1}}{n_{i+1}}}}, \\
s_{i+1} &= \sum_{i+1}^{k=1} r_k,
\end{aligned}$$

where $n_1 = n_G$, $\bar{d}_1 = \bar{d}_G$, $r_1 = r(G)$ and $s_1 = r_1$ for the original graph $G$. Before proceeding, we show basic properies of these sequences.

**Lemma 1.** *$n_i \geq \bar{d}_i$ and $r_i \geq \frac{\bar{d}_i}{2}$ hold for each $i$.*

*Proof.* Consider removing $k$ vertices from a graph $G$ and let $G'$ be the resulting graph. Then the decrease of the sum of the degrees, i.e., $\bar{d}_G n_G - \bar{d}_{G'} n_{G'}$ is at most $\sum_{j=1}^{k} deg(j)$. Since $G'$ has $n - k$ vertices, the sum of the degrees of $G'$ is at most $(n-k)^2$. Therefore, the sum of the degrees of $G$ is upper-bounded by $\sum_{j=1}^{k} deg(j) + (n-k)^2$, which means

$$\bar{d}n \leq \sum_{j=1}^{k} deg(j) + (n-k)^2.$$

And this leads to

$$n - k \leq \frac{\bar{d}n - \sum_{j=1}^{k} deg(j)}{n-k}.$$

Setting $k = s_i$, we can prove $n_i \geq \bar{d}i$.

Next we show that $r_i \geq \frac{\bar{d}_i}{2}$ for each $i$. This can be shown easily by the definition of $r_i$ and $n_i \geq \bar{d}i$.

□

One can see that the algorithm follows the basic idea given in the previous section almost exactly. $H(G)$ is the set of "high-degree" vertices. $\mathcal{V}_G$ is the family of $2(\log n)^2 + 1$ vertex sets. In

the previous overview of the algorithm, we consider the whole vertex set $V$ in $\mathcal{V}_G$ as a candidate to be a part of $C$, remove it from $G$ and go to the next level. Actually we select $r_i$ vertices from $V$ uniformly at random and remove only those $r_i$ ones from $G$. This is just for simpler analysis, in other words, it is enough for achieving our approximation factor to remove this number of vertices. One can easily see that each $C_j$ in line 7 is a vertex cover of $G$ at that level. If the level is $t$, then we directly obtain a vertex cover of $G$ by using $\mathcal{VC}_2$. Since each computation path splits into $(2(\log n)^2+1)^t$ paths at each level, we have $(2(\log n)^2+1)^t$ computation paths in total. Apparently DVC-Apx$(t,0,G)$ gives a correct vertex cover of the original graph $G$. In the next section, we analyze its size.

## 4 Analysis of Approximation Factor

In this section we evaluate the approximation factor of DVC-Apx described in the previous section. Actually, we prove the following theorem.

**Theorem 1.** *Given a graph $G$, whose average degree is $\overline{d} = \overline{d}_G$ and the maximum degree is $\Delta = \Delta_G$, DVC-Apx yields an approximate solution for $G$ whose approximation factor is at most*

$$\frac{2}{1+\{1-(1-\frac{\Delta}{2n})^t\}\gamma(G)}$$

*in time $O\left((\log n)^{2t}\right)$.*

Especially, since $(1-\frac{1}{x})^x < 1/e$, by setting $t = O(n/\Delta)$, the approximation factor obtained by DVC-Apx can be arbitrarily close to $\frac{2}{1+\gamma(G)/n}$. The correctness of the computation time is obvious from the recursion structure given in the previous section, which is $O((\log n)^{O(n/\Delta)})$ under the same setting. Thus we have the following corollary.

**Corollary 1.** *Suppose that $\Delta = \Omega(n\frac{\log\log n}{\log n})$. Then our algorithm runs in polynomial time and its approximation factor is $\frac{2}{1+\gamma(G)/2}$.*

In the following we focus on the analysis of the approximation factor, for which we need several lemmas.

**Lemma 2.** *Let $G=(V,E)$, $C \subseteq V$ be one of the minimum vertex cover of $G$ and $W$ be a set of vertices s.t. $|W| = n_1+n_2$, $|W\cap C| \geq n_1$ and $|W-C| \leq n_2$. Then we can construct a vertex cover whose approximation factor is at most $\frac{2}{1+\frac{n_1-n_2}{n}}$.*

This lemma is a generalization of Lemma 4.1 in [10]. In [10], the case that $n_2 = 0$ ($W \subseteq C$) was considered and the Lemma 2 can be proved similarly. Hence we omit the proof here. By using this lemma, we can concentrate only on finding a set of vertices which has a large fraction of vertices in $C$.

Consider the recursion tree generated by the execution of DVC-Apx on $G$. For a path $P$ from the root to a leaf in this tree, let $G_i^P$ be the input graph at the $i$th level (see Figure 2). We denote by $W_i^P$ the set of removed vertices at level $i$ on $P$ and let $W_P = \bigcup_{i=1}^t W_i^P$. Note that $\sum_{i=1}^t |W_i^P| = \sum_{i=1}^t r_i$ and $G_{i+1}^P = G_i^P - W_i^P$ holds for each $i$. In the rest of the paper, when we say a "node" in the tree, it often means the input graph in that node. Note that the root of tree is $G_1^P = G$. We often omit the superscript $P$ from these notations when the underlying path $P$ is clear from the context.

**Lemma 3.** *For any path $P$ and any level $i$ we have $n_{G_i} = n_i, \overline{d}_{G_i} \geq \overline{d}_i$ and $r(G_i) \geq r_i$.*

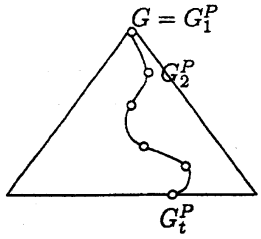


Figure 2: a recursion tree

*Proof.* By induction on $i$. For $i = 1$, these equalities hold by the definitions. For $i \geq 2$, since $G_{i+1} = G_i - W_i$, we have $n_{G_{i+1}} = n(G_i) - r_i = n_i - r_i = n_{i+1}$. Hence $n_{G_{i+1}} = n_{i+1}$. Since the average degree of $G_{i+1}$ is at least $\frac{\bar{d}_G n_G - 2\sum_{k=1}^{s_i} deg(k)}{n_{G_{i+1}}}$, we have

$$\begin{aligned} \bar{d}_{G_{i+1}} &\geq \frac{\bar{d}_G n_G - 2\sum_{k=1}^{s_i} deg(k)}{n_{i+1}} \\ &= \bar{d}_{i+1}. \end{aligned}$$

Therefore, $\bar{d}_{G_{i+1}} \geq \bar{d}_{i+1}$ holds. For $\{r_i\}$, using $n_{G_{i+1}} = n_{i+1}$ and $\bar{d}_{G_{i+1}} \geq \bar{d}_{i+1}$, we have

$$\begin{aligned} r(G_{i+1}) &= \frac{\bar{d}_{G_{i+1}}}{1 + \sqrt{1 - \frac{\bar{d}_{G_{i+1}}}{n_{G_{i+1}}}}} \\ &= \frac{\bar{d}_{G_{i+1}}}{1 + \sqrt{1 - \frac{\bar{d}_{G_{i+1}}}{n_{i+1}}}} \\ &\geq \frac{\bar{d}_{i+1}}{1 + \sqrt{1 - \frac{\bar{d}_{i+1}}{n_{i+1}}}} \\ &= r_{i+1}, \end{aligned}$$

which proves the lemma. □

**Lemma 4.** *Let $P$ be any path from the root to a leaf in the recursion tree and $G_1, G_2, \ldots, G_{t-1}$ be the nodes along $P$. For every $G_i$, with probability at least $1 - \frac{1}{n^2}$, the family $\mathcal{V}_{G_i}$ contains at least one set $V_j$ s.t. $|V_j \cap C| \geq (1 - 1/\log n)|V_j|$.*

*Proof.* If $|H(G_i) \cap C| \geq (1 - 1/\log n)|H(G_i)|$, then the claim holds with probability 1 since $H(G_i) \in \mathcal{V}_{G_i}$ and $H(G_i)$ satisfies the conditions. Otherwise, i.e., if $|H(G_i) \cap C| < (1 - 1/\log n)|H(G_i)|$, then the set $U_{G_i}$ of randomly chosen vertices contains a vertex $v$ s.t. $v \notin C$ with high probability. Indeed, when $|H(G_i)| \geq 2(\log n)^2$, the probability that $U_{G_i}$ contains such a vertex is:

$$\begin{aligned} \Pr[\exists v \in U_{G_i} - C] &\geq 1 - \prod_{v \in U_{G_i}} \Pr[v \notin U_{G_i} - C] \geq 1 - (1 - 1/\log n)^{2(\log n)^2} \\ &\geq 1 - \left(\frac{1}{e}\right)^{2\log n} = 1 - \frac{1}{n^2}. \end{aligned}$$

If $|H(G_i)| < 2(\log n)^2$, this probability is 1 since $U_{G_i} = H(G_i)$ and $|H(G_i) - C| > |H(G_i)|/\log n > 0$. For such a vertex $v \notin C$, $N_{G_i}(v) \subseteq C$, which means $|N_{G_i}(v) \cap | = |N_{G_i}(v)| > (1 - 1/\log n)|N_{G_i}(v)|$. Thus the claim holds. □

**Lemma 5.** *For any computation path $P$ of height $t$, the sum of the degree of the removed vertices, i.e., $\sum_{k=1}^{s_t} deg(k)$, is at least $(1-(1-\frac{\Delta}{2n})^t)\bar{d}n$.*

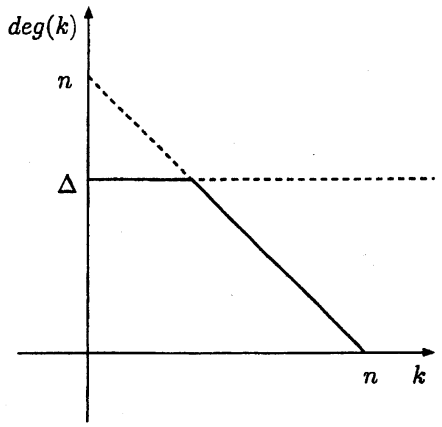


Figure 3: $deg(k)$

*Proof.* Let $a_i = \bar{d}n - 2\sum_{k=1}^{s_i} deg(k)$, which intuitively represents the lower bound of the number of remaining edges. We first prove that $a_i$ decreases exponentially. To do so, we show the following inequality.

$$\frac{a_{i+1}}{a_i} \leq 1 - \frac{\Delta}{2n}. \tag{1}$$

Supposing this inequality holds, we can show that $a_i \leq (1-\frac{\Delta}{2n})^i a_0 = (1-\frac{\Delta}{2n})^i \bar{d}n$ and the lemma will be proved. Hence from here we will prove equation (1).

First, $a_i - a_{i+1} = 2\sum_{k=s_i+1}^{s_{i+1}} deg(k)$ can be lower-bounded by

$$r_{i+1}\frac{deg(s_i)+deg(s_{i+1})}{2}. \tag{2}$$

This can be seen in Figure 4 because $\sum_{k=s_i+1}^{s_{i+1}} deg(k)$ equals to the gray area and expression (2) equals to the underlying trapezium.

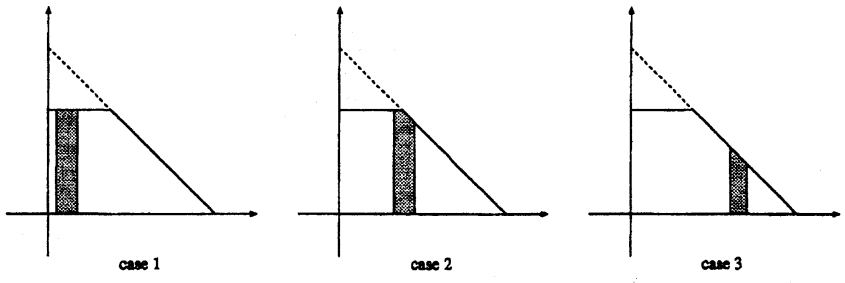


Figure 4: gray area represents $\sum_{k=s_i+1}^{s_{i+1}} deg(k)$

**case 1.** The case $s_i < s_{i+1} \leq n - \Delta$, it follows that $deg(s_i) = deg(s_{i+1}) = \Delta$. Therefore,

$$\sum_{k=s_i+1}^{s_{i+1}} deg(k) \geq r_{i+1}\frac{\Delta+\Delta}{2}$$

$$
\begin{aligned}
&= r_{i+1}\Delta \\
&\geq \frac{\overline{d}_{i+1}}{2}\Delta \\
&= \frac{\overline{d}n - 2\sum_{k=1}^{s_i} deg(k)}{n_{i+1}} \cdot \frac{\Delta}{2} \\
&= \frac{a_i}{n_{i+1}}\frac{\Delta}{2} \\
&\geq \frac{a_i}{2}\frac{\Delta}{n}.
\end{aligned}
$$

Here we used Lemma 1 and the definition of $\overline{d}_i$.

**case 2.** The case $s_{i+1} > n - \Delta \geq s_i$, $deg(s_i) = \Delta$ and $deg(s_{i+1}) = n - s_{i+1}$. Therefore,

$$
\begin{aligned}
\sum_{k=s_i+1}^{s_{i+1}} deg(k) &\geq r_{i+1}\frac{\Delta + n - s_{i+1}}{2} \\
&\geq \frac{a_i}{2}\frac{\Delta + n - s_{i+1}}{2n_{i+1}} \\
&\geq \frac{a_i}{2}\left(\frac{\Delta}{2n_{i+1}} + \frac{n - s_i - r_{i+1}}{2n_{i+1}}\right) \\
&\geq \frac{a_i}{2}\left(\frac{\Delta}{2n_{i+1}} + \frac{n_{i+1} - r_{i+1}}{2n_{i+1}}\right) \\
&\geq \frac{a_i}{2}\frac{\Delta}{2n_{i+1}} \\
&\geq \frac{a_i}{2}\frac{\Delta}{2n}.
\end{aligned}
$$

Here we used $n_{i+1} = n - s_i$ and $n_{i+1} \geq r_{i+1}$.

**case 3.** Finally, the case $s_{i+1} > s_i > n - \Delta$, $deg(s_i) = n - s_i$ and $deg(s_{i+1}) = n - s_{i+1}$. Thus

$$
\begin{aligned}
\sum_{k=s_i+1}^{s_{i+1}} deg(k) &\geq r_{i+1}\frac{n - s_{i+1} + n - s_i}{2} \\
&\geq \frac{a_i}{2}\frac{2n - s_{i+1} - s_i}{2n_{i+1}} \\
&= \frac{a_i}{2}\frac{2(n - s_i) - s_{i+1} + s_i}{2n_{i+1}} \\
&= \frac{a_i}{2}\left(1 - \frac{r_{i+1}}{2n_{i+1}}\right) \\
&\geq \frac{a_i}{4}.
\end{aligned}
$$

Here we used $r_i \leq n_i$.

Thus in any case $a_i - a_{i+1} = \sum_{k=s_i+1}^{s_{i+1}} deg(k) \geq \frac{a_i}{2} \cdot \frac{\Delta}{2n}$ holds since $\frac{\Delta}{2n} = \min\left\{\frac{\Delta}{n}, \frac{\Delta}{2n}, \frac{1}{2}\right\}$ Therefore it follows that $a_{i+1} \leq (1 - c)a_i$ and we have shown that $a_i$ decreases exponentially. □

Using this lemma, we next show that $s_i$ converges to $\gamma(G)$ and the difference $\gamma(G) - s_i$ also decreases exponentially.

**Lemma 6.** *For any computation path $P$ of height $t$, the sum of the number of removed vertices, i.e., $|W_P| = \sum_{k=1}^{t} r_k$, is at least $(1 - (1 - \frac{\Delta}{2n})^t)\gamma(G)$, where $c = \min\{\frac{\Delta}{2n}, \frac{1}{2}\}$.*

*Proof.* In Lemma 5, we have shown that $2\sum_{k=1}^{s_i} deg(k)$ converges to $\bar{d}n$ exponentially. This implies that $s_i$ converges to the value of the solution of the following equation:

$$2\sum_{k=1}^{x} deg(k) = \bar{d}n. \tag{3}$$

So let us first solve this equation. To solve this, we have to consider two cases. One case is the case $1 - \frac{\Delta}{n} \geq \frac{\bar{d}}{2\Delta}$, in which $deg(k) = \Delta$ holds for all $k$. This can be shown as follows. Let $x_0$ be the solution of the above equation and suppose that $deg(k) = n - k$ holds for some $k \leq x_0$. Then it follows that $x > n - \Delta \geq \frac{\bar{d}n}{2\Delta}$ and $\bar{d}n = 2\sum_{k=1}^{x} deg(k) > 2\sum_{k=1}^{\frac{\bar{d}n}{2\Delta}} deg(k) = \bar{d}n$. This is a contradiction and $deg(k) = \Delta$ holds for all $k$.

The other case is $1 - \frac{\Delta}{n} < \frac{\bar{d}}{2\Delta}$, in which for some $k$ it holds that $deg(k) = n - k$. This can be shown similarly.

In the first case, i.e. $1 - \frac{\Delta}{n} \geq \frac{\bar{d}}{2\Delta}$,

$$2\sum_{k=1}^{x} deg(k) = 2x\Delta$$

holds and we obtain $x = \frac{\bar{d}n}{2\Delta}$ by solving (3). In the second case, $1 - \frac{\Delta}{n} < \frac{\bar{d}}{2\Delta}$, we have

$$2\sum_{k=1}^{x} deg(k) = 2\Delta(n-\Delta) + 2\frac{(\Delta + (n-x))(x-(n-\Delta))}{2}.$$

This can be seen by the gray area of Figure 5. Solving (3) by using the above equality, we have $x = n - \sqrt{\Delta(2n-\Delta) - \bar{d}n}$. After all, we have shown that $s_i$ converges to $\gamma(G)$.

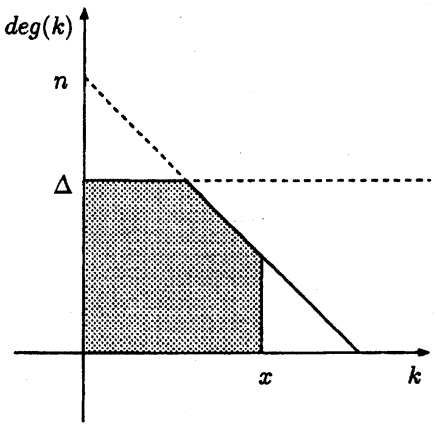


Figure 5: the gray area represents $\sum_{k=1}^{x} deg(k)$

Next we will show that the rate of convergence is also $\left(1 - \frac{\Delta}{2n}\right)$. To show this, we prove here that the following inequality holds for each $i$.

$$\sum_{k=1}^{s_i} deg(k) \geq \frac{\bar{d}n}{\gamma(G)} s_i \tag{4}$$

Supposing this inequality, we have $s_i \geq (1 - (1-c)^i)\gamma(G)$ and the lemma will be proved.

This can be easily shown by seeing Figure 6 since the upper curve represents $\sum_{k=1}^{s_i} deg(k)$ and the lower line represents $\frac{\bar{d}n}{\gamma(G)}s_i$. For $0 \leq x \leq n-\Delta$, $\sum_{k=1}^{x} deg(k) = \Delta x \geq \frac{\bar{d}n}{\gamma(G)}x$ holds by definition of $\gamma(G)$ For $x > n-\Delta$, $\sum_{k=1}^{x} deg(k) = \Delta(n-\Delta) + (\Delta + (n-x))(x-(n-\Delta))/2$ and let $f(x)$ be the right hand side of the equation. Since $f(x)$ is convex upward and $f(x) > \frac{\bar{d}n}{\gamma(G)}x$ holds for $x = n-\Delta, \gamma(G)$, we have $f(x) > \frac{\bar{d}n}{\gamma(G)}x$ for all $n-\Delta \leq x \leq \gamma(G)$. Hence equation (4) holds and the lemma proved.

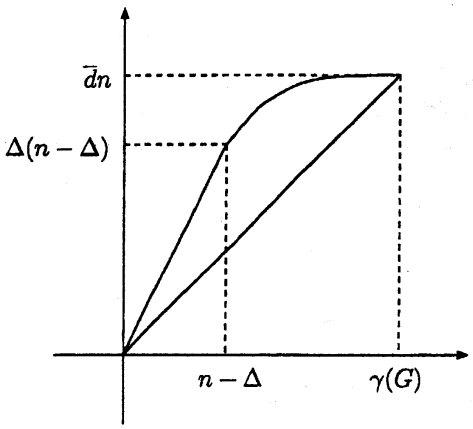


Figure 6:

□

**Lemma 7.** *For every $V_j \in \mathcal{V}_{G_i}$, $|V_j| \geq r_i$.*

*Proof.* Recall that $H(G) = \{v \mid d(v) \geq r(G)\}$. For each $v \in U_{G_i}$, therefore, we have $d(v) \geq r(G_i)$ and $|N_{G_i}(v)| \geq r(G_i)$. By lemma 3, $r(G_i) \geq r_i$ holds, which also means $|N_{G_i}(v)| \geq r_i$. To prove $|H(G_i)| \geq r(G_i)$, suppose for contradiction that $|H(G_i)| < r(G_i)$. This means only $|H(G_i)|$ vertices have a degree at most $n_{G_i}$ and $n_{G_i} - |H(G_i)|$ vertices have a degree less than $r(G)$. Now the total sum of the degree of vertices in $G_i$ is bounded by

$$\begin{aligned}
\sum_{v \in V(G_i)} d_{G_i}(v) &< |H(G_i)|n_{G_i} + (n_{G_i} - |H(G_i)|)r(G_i) \\
&= n_{G_i}r(G_i) + (n_{G_i} - r(G_i))|H(G_i)| \\
&< n_{G_i}r(G_i) + (n_{G_i} - r(G_i))r(G_i) \\
&= (2n_{G_i} - r(G_i))r(G_i).
\end{aligned}$$

Here the second inequality holds since $r(G_i) \leq \bar{d}_{G_i} \leq n_{G_i}$. Since $\bar{d}_{G_i}n_{G_i} = \sum_{v \in V(G_i)} d_{G_i}(v)$, it follows that $\bar{d}_{G_i}n_{G_i} < (2n_{G_i} - r(G_i))r(G_i)$. Solving this inequality we obtain $r(G) < \frac{\bar{d}_{G_i}}{1+\sqrt{1-\bar{d}_{G_i}/n_{G_i}}}$, which contradicts the definition of $r(G)$. □

By this lemma we can always select $r_i$ vertices in the 7th line of the algorithm. Hence $|W_P| = \sum_{i=1}^{t} r_i$ for any path $P$.

**Lemma 8.** *Let $C$ be one of the minimum vertex covers of $G$. Then, with high probability, there is a path $P$ s.t. $|W_P \cap C|$ is at least $(1-o(1))\sum_{i=1}^{t} r_i$.*

*Proof.* We call a path $P$ a *good path* if, at every node $G_i^P$, $\mathcal{V}_{G_i^P}$ has a set $V_j$ s.t. at least $1-\frac{1}{\log n}$ fraction of $V_j$ is in $C$. Recall that for some set $V_j' \subseteq V_j$ of randomly chosen vertices s.t. $|V_j'| = r_i$, $G_{i+1}^P = G_i^P - V_j'$ holds.

We can observe that, by Lemma 7, the probability that a recursion tree has at least one good path of height $t$ is at least $(1-\frac{1}{n^2})^t$ and $t$ is obviously at most $n$ since $|G_i| > |G_{i+1}|$ holds for any $i$. Thus this probability is at least $(1-\frac{1}{n^2})^n$. Note that since $(1-\frac{1}{x})^x \geq 1/2e$ for sufficiently large $x$, the probability is greater than $(1/2e)^{1/n}$, which is almost equal to 1 for sufficiently large $n$.

Next let $P$ be a good path on the tree generated by the algorithm and we consider the probability that $|W_P \cap C| > (1-1/\log n)\sum_{i=1}^t r_i$. Let $W_P = \{v_1, v_2, \ldots, v_{|W_P|}\}$ be the set of all the removed vertices along path $P$ and $X_i$ be the random variable s.t. $X_i = 1$ if $v_i \in C$ and $X_i = 0$ otherwise. Then $X = \sum_{i=1}^{|W_P|} X_i$ is a random variable that represents $|W_P \cap C|$. The expectation $\mu$ of $X$ is

$$\mu = E[X] = E\left[\sum_{i=1}^{|W_P|} X_i\right] = \sum_{i=1}^{|W_P|} E[X_i] \geq \left(1-\frac{1}{\log n}\right)\sum_{i=1}^{t} r_i.$$

This is because each vertices in $W_i^P$ is randomly chosen from some $V_j$ which has at least $1-1/\log n$ fraction of vertices in $C$. Next we estimate the probability that $X$ deviates from its expectation. Since $X_1, X_2, \ldots, X_{|W_P|}$ can be seen as independent Poisson trials, we can use Chernoff Bound to imply that

$$\Pr[X < (1-1/\log n)\,\mu] < e^{-\frac{\mu}{2(\log n)^2}} < e^{-\frac{(1-1/\log n)\overline{d}}{2(\log n)^2}} = e^{-O\left(\frac{\overline{d}}{(\log n)^2}\right)}.$$

Here, the second inequality follows from $\mu \geq (1-\frac{1}{\log n})\sum_{i=1}^t r_i \geq (1-\frac{1}{\log n})r_1 \geq \frac{(1-\frac{1}{\log n})\overline{d}}{2}$ by Lemma 6. Hence the probability we are considering is at least:

$$\left(1-\frac{1}{n^2}\right)^t \cdot \left(1-e^{-O\left(\frac{\overline{d}}{(\log n)^2}\right)}\right).$$

Since we are now considering (almost) dense graphs, this probability is almost equal to 1 for sufficiently large $n$. Thus the lemma follows. □

Putting these lemmas all together, we can now prove the theorem. By Lemma 8, with high probability, there is a path $P$ s.t. $|W_P \cap C| \geq (1-o(1))|W_P|$, which means $|W_P - C| \leq o(1)|W_P|$. Then by Lemma 2 the approximation factor of DVC-Apx is at most

$$\frac{2}{1+\frac{(1-o(1))|W_P|-o(1)|W_P|}{n}} = \frac{2}{1+\frac{(1-o(1))|W_P|}{n}}.$$

Next, by Lemma 6, $|W_P|$ is at least $(1-(1-\frac{\Delta}{2n})^t)\gamma(G)$. Therefore we have

$$\frac{2}{1+\frac{(1-o(1))|W_P|}{n}} \leq \frac{2}{1+(1-o(1))\{1-(1-1-\frac{\Delta}{2n})^t\}\gamma(G)}.$$

That completes the proof of Theorem 1.

## 5 Concluding remarks

There are a few remarks we could not mention so far: (i) Although we have mainly discussed the case that $\Delta = \Omega(n\frac{\log\log n}{\log n})$, the algorithm works for smaller $\Delta$ if we allow super-polynomial running time. Especially, if $\Delta = \omega(\log\log n)$, then the algorithm achieves the same approximation factor (i.e., strictly less than two if $\overline{d}/\Delta$ is constant) in time $2^{o(n)}$. (ii) The approximation factor is strictly less than two only if $\overline{d}/\Delta$ is constant. This condition can be slightly relaxed as follows: Namely,

suppose that we can obtain a graph $G'$ such that $\bar{d}_{G'}/\Delta_{G'}$ is constant by removing $o(n)$ vertices from the original graph $G$. Then we can apply our algorithm to $G'$ and can obtain an answer (a vertex cover) for the original $G$ by adding the $o(n)$ removed vertices to the answer for $G'$. One can see easily that this still gives us a factor of less than two.

## References


[1] Sanjeev Arora, David Karger, and Marek Karpinski. Polynomial time approximation schemes for dense instances of $\mathcal{NP}$-hard problems. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 284–293, Las Vegas, Nevada, 29 May–1 June 1995.

[2] Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. Assoc. Comput. Mach.*, 41:153–180, 1994.

[3] Piotr Berman and Toshihiro Fujito. On the approximation properties of independent set problem in degree 3 graphs. In *Workshop on Algorithms and Data Structures*, pages 449–460, 1995.

[4] Clementi and Trevisan. Improved non-approximability results for minimum vertex cover with density constraints. *TCS: Theoretical Computer Science*, 225, 1999.

[5] Irit Dinur and Shmuel Safra. The importance of being biased. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC-02)*, pages 33–42, New York, May 19–21 2002. ACM Press.

[6] Magnús M. Halldórsson and Jaikumar Radhakrishnan. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 439–448, Montréal, Québec, Canada, 23–25 May 1994.

[7] Eran Halperin. Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. *SIAM Journal on Computing*, 31(5):1608–1623, October 2002.

[8] Johan Hastad. Some optimal inapproximability results. *Electronic Colloquium on Computational Complexity (ECCC)*, 4(037), 1997.

[9] Hochbaum. Efficient bounds for the stable set, vertex cover and set packing problems. *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 6, 1983.

[10] Marek Karpinski and Alexander Zelikovsky. Approximating dense cases of covering problems. *Electronic Colloquium on Computational Complexity (ECCC)*, 4(004), 1997.

[11] S. Khot and O. Regev. Vertex cover might be hard to approximate to within $2-\varepsilon$. In *Proc. of 18th IEEE Annual Conference on Computational Complexity (CCC)*, pages 379–386, 2003.